|  |
| --- |
| **Table Of Contents**   1. **Introduction**  * Deep Learning * Key features of Deep Learning * Deep Learning Techniques  1. **CNN**  * Key concepts of CNN * Applications of CNN  1. **RNN**  * Key concepts of RNN * Applications of RNN  1. **Transformer**    * Key Components of Transformer    * Applications 2. **PyTorch**  * Features of Transformer |

**Name - Sania Shahid Sheikh**

**UGMR Roll no - UGMR20230031**

**Comprehensive Report on**

**Deep Learning**

**1. INTRODUCTION**

**What is Deep Learning?**

Deep learning is a subset of machine learning, which itself is a subset of artificial intelligence (AI). It is based on artificial neural networks, which are algorithms inspired by the structure and function of the brain. These neural networks consist of layers of nodes, or "neurons," that process data and can learn to perform tasks by considering examples, generally without being programmed with task-specific rules.

**Key features of deep learning include:**

1. **Neural Networks**: The core component of deep learning. These networks are composed of multiple layers:
   * **Input Layer**: Where the model receives the data.
   * **Hidden Layers**: Where the data is processed. These layers can be many, leading to the term "deep" learning.
   * **Output Layer**: Where the result is produced.
2. **Learning**: Deep learning models improve their performance by learning from large amounts of data. This process involves adjusting the weights and biases in the neural network to minimize the difference between the predicted output and the actual output.
3. **Backpropagation**: A technique used to train neural networks by adjusting the weights based on the error of the network's prediction compared to the actual result. This involves propagating the error back through the network and updating the weights to reduce the error.
4. **Activation Functions**: Functions that determine the output of a node. Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Tanh.
5. **Weights and Biases**:
   * **Weights**: Parameters that adjust the input signal strength. They are learned during training.
   * **Biases**: Constants added to the inputs to help the model fit the data better.
6. **Forward Propagation**:
   * The process of passing input data through the layers of the network to get the final output. Each layer applies weights, biases, and activation functions to the input from the previous layer.

**Types of neural Networks:**

 **Convolutional Neural Networks (CNNs)**: Typically used for image recognition and processing.

 **Recurrent Neural Networks (RNNs)**: Often used for sequential data like time series or natural language processing.

 **Generative Adversarial Networks (GANs)**: Used for generating new data samples similar to a given dataset.

 **Autoencoders**: Used for unsupervised learning tasks such as dimensionality reduction and feature learning.

**Deep Learning Techniques**

1. **Supervised Learning**: The model is trained on labeled data. Examples include image classification, speech recognition, and translation.
2. **Unsupervised Learning**: The model tries to learn the underlying structure of the data without labeled outputs. Examples include clustering and anomaly detection.
3. **Semi-Supervised Learning**: Combines a small amount of labeled data with a large amount of unlabeled data during training.
4. **Reinforcement Learning**: The model learns to make decisions by interacting with an environment and receiving rewards or penalties. This is often used in game playing and robotics.

**Applications of Deep Learning:**

1. **Computer Vision**: Image classification, object detection, facial recognition, and medical image analysis.
2. **Natural Language Processing (NLP)**: Language translation, sentiment analysis, text generation, and chatbots.
3. **Speech Recognition**: Converting spoken language into text, voice-activated assistants.
4. **Healthcare**: Predicting diseases, personalized treatment plans, drug discovery.
5. **Autonomous Vehicles**: Object detection, path planning, and decision-making for self-driving cars.

**2) CNN (Convolutional Neural Network):**

A Convolutional Neural Network (CNN) is a type of deep learning algorithm specifically designed for processing and analyzing visual data. CNNs have been particularly successful in tasks related to image and video recognition, classification, and segmentation. Here's an in-depth look at CNNs:

**2.1) Key Concepts of CNNs**

1. **Convolutional Layers**:
   * **Convolution Operation**: The fundamental building block of a CNN, which involves a filter (or kernel) sliding over the input data to produce a feature map. This operation captures local spatial information and patterns within the input data.
   * **Filters/Kernels**: Small matrices that are applied to the input data to detect features such as edges, textures, or patterns. Multiple filters are used to detect different features.
2. **Receptive Field**:
   * The region of the input data that a filter processes at a time. The size of the receptive field determines how much of the input the filter covers, capturing fine or coarse details.
3. **Activation Function**:
   * Applied to the output of the convolution operation to introduce non-linearity into the model, allowing it to learn more complex patterns. Common activation functions include ReLU (Rectified Linear Unit).
4. **Pooling Layers**:
   * **Purpose**: To reduce the spatial dimensions (width and height) of the feature maps while retaining important information. This helps in reducing the computational load and controlling overfitting.
   * **Types**: Max pooling (selecting the maximum value within a region) and average pooling (calculating the average value within a region).
5. **Fully Connected Layers**:
   * After several convolutional and pooling layers, the high-level reasoning in the neural network is done via fully connected layers. These layers connect every neuron in one layer to every neuron in the next layer.
   * **Flattening**: The process of converting the 2D matrix of feature maps into a 1D vector before feeding it into fully connected layers.
6. **Dropout**:
   * A regularization technique used to prevent overfitting. During training, a fraction of the neurons is randomly set to zero, forcing the network to learn more robust features.

**2.3) Applications of CNNs**

1. **Image Classification**: Recognizing objects within images (e.g., identifying whether an image contains a cat or a dog).
2. **Object Detection**: Detecting and locating objects within images (e.g., identifying and drawing bounding boxes around cars in an image).
3. **Image Segmentation**: Classifying each pixel in an image into different categories (e.g., segmenting a medical image into regions corresponding to different organs).
4. **Facial Recognition**: Identifying and verifying individuals based on their facial features.
5. **Medical Imaging**: Analyzing medical images (e.g., MRI, CT scans) to detect and diagnose diseases.
6. **Autonomous Vehicles**: Processing visual data to recognize and respond to traffic signs, pedestrians, and other vehicles.

**3) RNN (Recurrent Neural Network)**

A Recurrent Neural Network (RNN) is a type of neural network designed for processing sequential data, where the order of the data points is important. RNNs are particularly useful for tasks that involve time series data, natural language processing, and any other applications where the temporal or sequential aspect of the data is crucial. Here’s a detailed explanation of RNNs:

**3.1) Key Concepts of RNNs**

1. **Sequential Data**:
   * Unlike feedforward neural networks, which assume all inputs and outputs are independent of each other, RNNs have connections that create cycles in the network, allowing information to persist over time.
2. **Hidden State**:
   * The hidden state is a key feature of RNNs that captures information about previous inputs in the sequence. At each time step, the hidden state is updated based on the current input and the previous hidden state.
   * Mathematically, the hidden state hth\_tht​ at time step ttt is computed as: ht=σ(Whht−1+Wxxt+b)h\_t = \sigma(W\_h h\_{t-1} + W\_x x\_t + b)ht​=σ(Wh​ht−1​+Wx​xt​+b) where WhW\_hWh​ and WxW\_xWx​ are weight matrices, xtx\_txt​ is the input at time step ttt, bbb is a bias vector, and σ\sigmaσ is an activation function (typically tanh or ReLU).
3. **Output**:
   * The output yty\_tyt​ at each time step ttt is generated based on the hidden state: yt=σ(Wyht+c)y\_t = \sigma(W\_y h\_t + c)yt​=σ(Wy​ht​+c) where WyW\_yWy​ is the output weight matrix and ccc is a bias vector.

**3.2) Applications of RNNs**

1. **Natural Language Processing (NLP)**:
   * **Language Modeling**: Predicting the next word in a sentence.
   * **Machine Translation**: Translating text from one language to another.
   * **Text Generation**: Generating coherent text based on a given input.
   * **Speech Recognition**: Converting spoken language into text.
   * **Sentiment Analysis**: Determining the sentiment of a piece of text.
2. **Time Series Prediction**:
   * Forecasting future values based on past observations, such as stock prices, weather data, or sales figures.
3. **Sequence Classification**:
   * Classifying sequences of data, such as determining the genre of a piece of music or identifying activities in a video.
4. **Image Captioning**:
   * Generating descriptive captions for images by combining CNNs (for image processing) with RNNs (for sequence generation).

**4) TRANSFORMER**

The Transformer is a type of deep learning model introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017. It was designed to handle sequential data tasks more effectively than previous models, particularly in natural language processing (NLP). Here’s a breakdown of what makes transformers distinct:

**4.1) Key Components of Transformer**

**Self-Attention Mechanism**:

* **Self-Attention** allows the model to weigh the importance of different words (or tokens) in a sequence relative to each other. This mechanism helps the model focus on relevant parts of the input sequence when processing each token.
* **Scaled Dot-Product Attention** is a core part of self-attention, calculating the attention weights by taking the dot product of queries and keys, scaling it, and applying a softmax function.

**Multi-Head Attention**:

* Instead of having a single attention mechanism, transformers use multiple attention heads. Each head learns different aspects of the relationships between tokens, and their outputs are concatenated and linearly transformed.

**Positional Encoding**:

* Since transformers don’t inherently understand the order of tokens (as they process tokens in parallel), they use positional encodings to inject information about the position of each token in the sequence. This allows the model to take into account the sequential nature of the data.

**Encoder-Decoder Architecture**:

* **Encoder**: Consists of a stack of identical layers, each with two sub-layers – multi-head self-attention and a position-wise fully connected feed-forward network. The encoder processes the input sequence and produces a set of representations.
* **Decoder**: Also consists of a stack of identical layers but with an additional sub-layer for cross-attention, which attends to the encoder's output. The decoder generates the output sequence one token at a time, attending to both the previous tokens in the output and the encoder's representations.

**4.2) Applications of Transformer**

* **Natural Language Processing**: Language translation, text summarization, sentiment analysis, and more.
* **Time Series Forecasting**: Stock market prediction, weather forecasting.
* **Speech Processing**: Speech recognition, text-to-speech synthesis.
* **Image Processing**: Object detection, image captioning.

**5) PyTorch:**

PyTorch is an open-source machine learning library primarily developed by Facebook's AI Research lab (FAIR). It is widely used for developing and training deep learning models and provides a flexible and easy-to-use framework that supports dynamic computational graphs. Here’s a detailed overview of PyTorch:

**5.1) Key Features of PyTorch**

1. **Dynamic Computational Graphs**:

Unlike TensorFlow's static graphs, PyTorch uses dynamic graphs (define-by-run), which means the computation graph is built on-the-fly as operations are executed. This makes it easier to debug and allows for more flexible model building.

1. **Tensors**:

* Tensors are the fundamental data structures in PyTorch, similar to NumPy arrays but with support for GPU acceleration. They enable efficient computation on both CPUs and GPUs.
* PyTorch provides a comprehensive set of operations for tensors, including mathematical operations, indexing, slicing, and more.

1. **Automatic Differentiation**:

* PyTorch’s autograd module provides automatic differentiation for all operations on tensors. It tracks all operations on a tensor and allows for easy gradient computation, which is essential for backpropagation during training.

1. **Neural Network Library (torch.nn)**:

* PyTorch includes a high-level neural network library that provides building blocks for constructing neural networks, such as layers, loss functions, and optimizers.

1. **CUDA Support**:

* PyTorch integrates seamlessly with CUDA, NVIDIA’s parallel computing platform, allowing users to leverage GPUs for faster computation and training.

1. **Rich Ecosystem**:

* PyTorch has a vibrant ecosystem with libraries for various tasks, including:
  + **torchvision**: For computer vision tasks.
  + **torchaudio**: For audio processing tasks.
  + **torchtext**: For natural language processing (NLP) tasks.

**CNN Image Classification Model GitHub link:-** [**https://github.com/saniaaa111/FMML\_Project\_and\_Labs/blob/main/CNN\_ImageClassification.ipynb**](https://github.com/saniaaa111/FMML_Project_and_Labs/blob/main/CNN_ImageClassification.ipynb)

**RNN Time Series Forecasting Model GitHub Link:-**

[**https://github.com/saniaaa111/FMML\_Project\_and\_Labs/blob/main/RNN\_TimeSeriesForecasting.ipynb**](https://github.com/saniaaa111/FMML_Project_and_Labs/blob/main/RNN_TimeSeriesForecasting.ipynb)

**Transformer based Text Classification Model GitHub Kink:-**

[**https://github.com/saniaaa111/FMML\_Project\_and\_Labs/blob/main/Transformer\_model.ipynb**](https://github.com/saniaaa111/FMML_Project_and_Labs/blob/main/Transformer_model.ipynb)